Physical computing from Scratch using scratchClient – **Advanced**

Control servos, LEDs and more from Scratch using RPi, Arduino, scratchClient

Need to be updated for Scratch 2

Hans de Jong & Gerhard Hepp

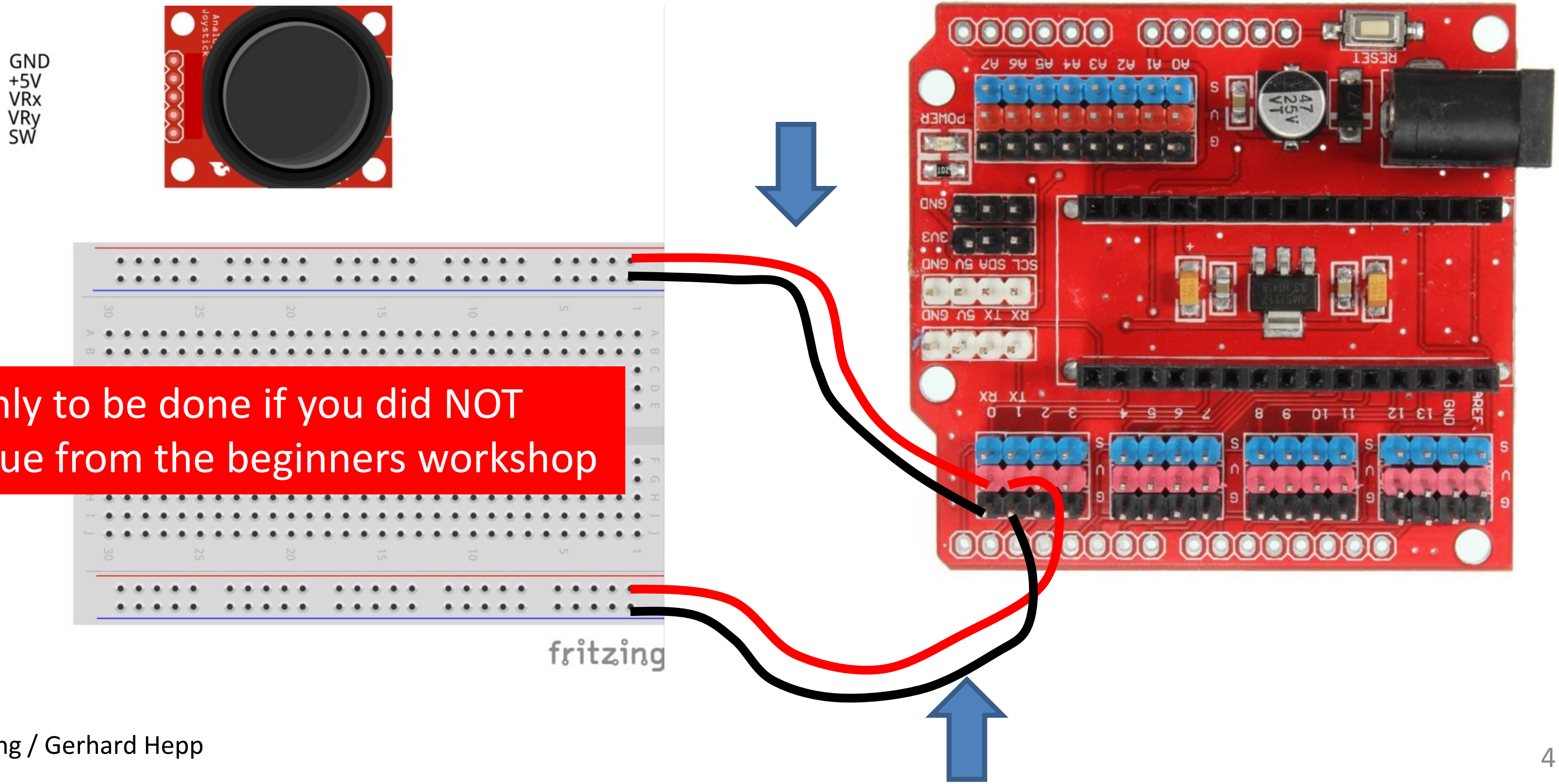
Workshop for the Pi And More and Maker Faire conferences

Before we start

If you do not roll over from the intermediate workshop …

- If you are familiar with scratchClient, but not with the new config tool …
  - … you may want to look at some slides of the beginners workshop
- Other than that, you do **not** have to build up the complete setup what the people of the beginners workshop did.
  - Only those things on the next slide.
- You will however see the components of the beginners workshop in the diagrams.
  - Just ignore those.

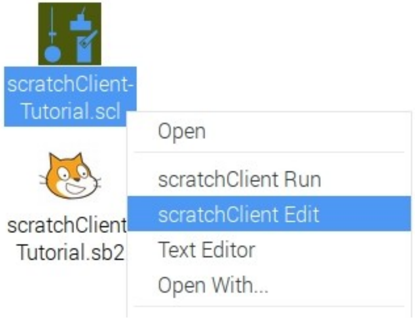
Connect the power wires



Only to be done if you did NOT continue from the beginners workshop

Make the config file for all steps

- Open config file by right clicking and choose scratchClient Edit
- In order not to loose time, it is recommended to add all setups in the config file now.
  - If you continued from the beginners workshop then leave in what you have, only add.
  - D3: out, pwm, 3Color Red LED
  - D5: out, pwm, 3Color Blue LED
  - D6: out, pwm, 3Color Green LED
  - D8: in, counter_pullup, Counter button
  - D10: in, counter, IR sensor



- A0: in, analog, Rain Sensor
- A1: in, input_pullup, Tilt Sensor
- A2: out, output, Relay
- A3: in, analog, Sound Sensor
- A5: in, input_pullup, Joystick Button
- A6: in, analog, JoyStickY
- A7: in, analog, JoyStickX

Hans de Jong / Gerhard Hepp

Choose the topics you want to give priority

- The advanced workshop may be too short to do all activities.
  - So pick the order of the topics from the list
- The red topics teach you more about scratchClient.
- The green topics teach you more about electronics, sensors and engineering.
- If you rather would do pieces of the beginners workshop then that is fine as well.

(This) Advanced workshop

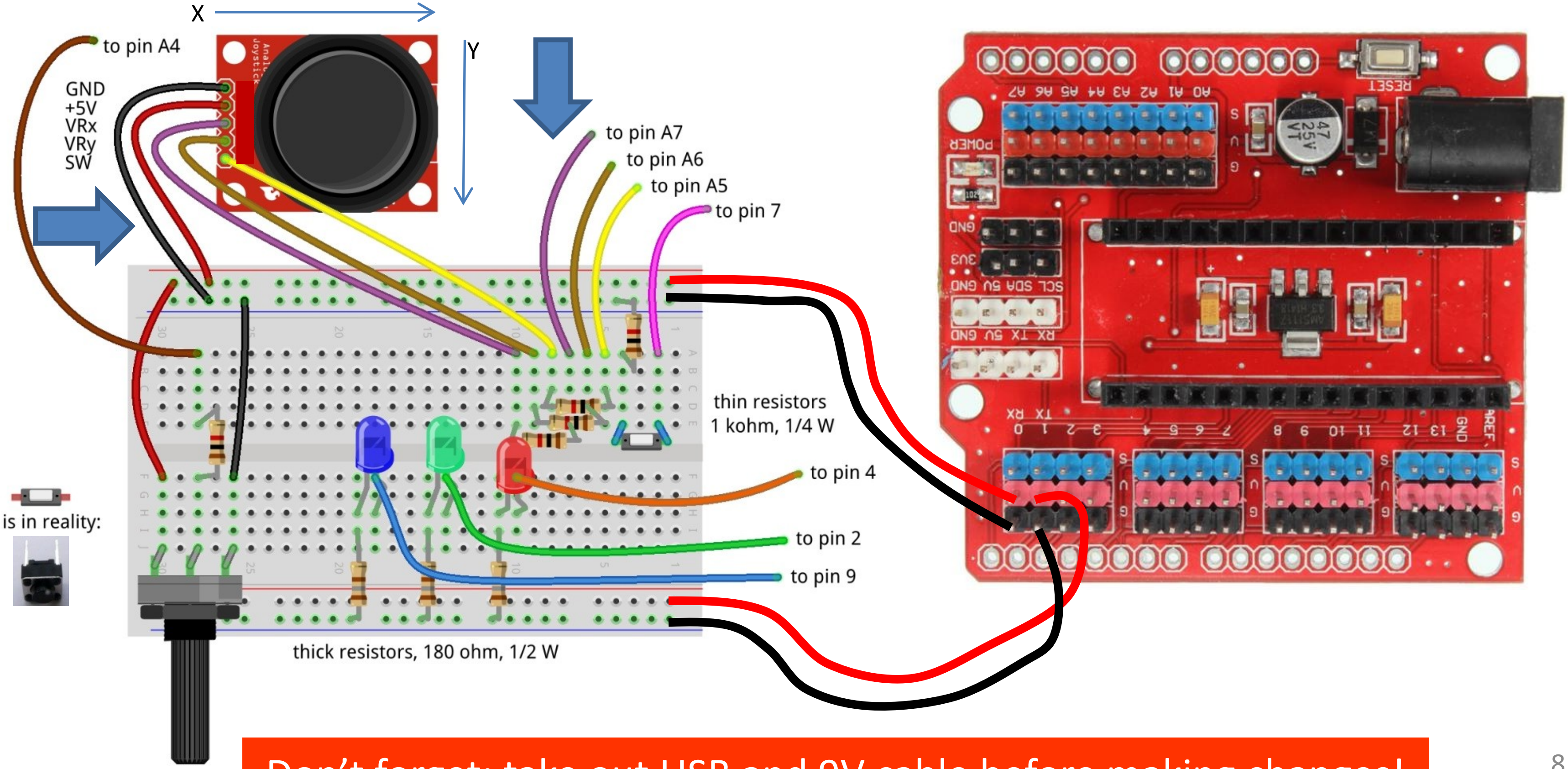
- Joystick
  - Control position or control speed
  - Take care of calibration and drift
- 3-color LED
  - To make any color
- Counter function
  - With button or IR slotted sensor
- Rain sensor
- Tilt sensor
- Sound sensor

Expert workshop

- Strongly link the config file to the Arduino
  - For safety
  - For connection errors
- Control multiple Arduinos concurrently
- Power On Self Test
- Controlling a relay
- Controlling 220 Volt appliances (only info)
- Controlling a camera (only info)

Part 1: Joy stick

Adding the joy stick



thin resistors 1 kohm, 1/4 W

thick resistors, 180 ohm, 1/2 W

is in reality:

Don’t forget: take out USB and 9V cable before making changes!

Try it out

- A joy stick has two small potentiometers and a button
  - When the joy stick is released (neutral position), the potentiometers are in the middle of the value range (middle between 0 and 1024)
  - There is one potentiometer for X and one for Y
  - The button is operated when pressing the button.
- Reconnect and repower
- Wait till the Arduino LED blinks slowly.
- Now try moving and pressing the joy stick. Look at the 3 joy stick values to change
  - JoystickX and JoystickY between about 0 and about 1023.
  - Joystick Button between 0 and 1

Uses of Joysticks

- You can directly use the value of the joystick
  - E.g. to control the position of a servo
  - E.g. to control the intensity of a LED
  - When releasing the knob, it will move back to the middle value.
  - E.g.: LED = JoystickX
- You can alternatively use the joystick to determine the speed of the change
  - E.g. move a servo fast or slow. Let the servo stop at the latest position when you release the knob.
    - E.g.: LED = LED + (JoystickX – 512) / 200 (512 = neutral position value, 200 = sensitivity)
  - You can also use the Scratch pen function draw on the screen.
    - And e.g. use the potentiometer to change pen width or color.
- Take care of drift and jitter (see next slide)

Take care of drift and jitter

- Not all joysticks will give the same value if they are in the middle position
- Influenced by temperature, the value produced in the middle position can drift over time.
- Therefore build in some threshold around the middle position
  - If the middle position is 512, then do only react if the value changes by at least ca. 15, so e.g. > 525 or < 500.
  - Whether 15 is enough as threshold you will learn over time. Increase the value if it drifts more than that.
  - If you have a general program that works with several joysticks then you may have to use a larger threshold, or you need to calibrate (adapt the program for each particular servo).

Part 2: Control a 3-color LED

To make all colors

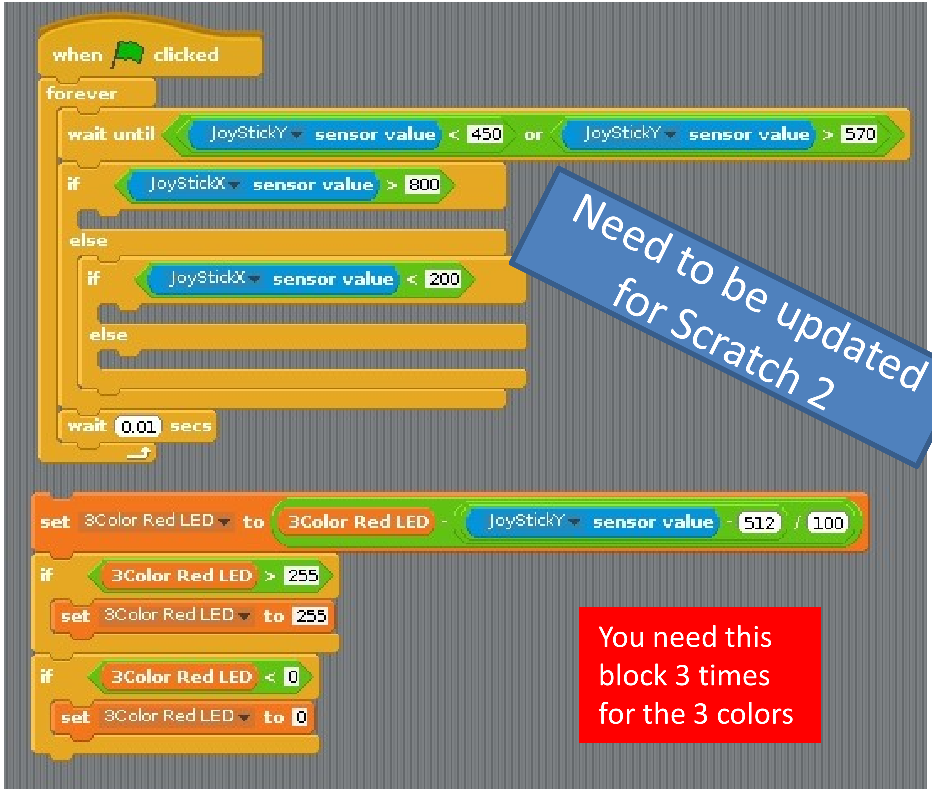
Control a 3-color LED

- A 3-color LED has 3 LEDs in one package
  - Green
  - Red
  - Blue
- Use PWM to change the intensity of each color
  - In that way you can create the entire spectrum of light
  - Including white light
- Connect the 3-color LED carefully to the pins as defined in the config file.
- Also connect the minus (–) pin to GND
- Reconnect and repower
- No need to restart scratchClient (since no update to the config file was made)
- Give the corresponding variables in Scratch values between 0 and 100 with the slider on the variable.
  - Does it work?
- We will see in the next slide how to use values between 0 and 255

Don’t forget: take out USB and 9V cable before making changes!

Connecting the 3-color LED to the joy stick (via Scratch)

- Write a program in Scratch that:
  - Uses the value of the X direction of the joy stick to determine red, green or blue
  - Uses the value of the Y direction to change the intensity of the respective LED
  - When releasing the joy stick, the colors should stay
  - You will need these elements



```
when [flag] clicked
forever
    wait until <<(JoyStickY sensor value) < 450> or <(JoyStickY sensor value) > 570>>
    if <(JoyStickX sensor value) > 800>
    else
        if <(JoyStickX sensor value) < 200>
        else
    wait 0.01 secs

set 3Color Red LED to ((3Color Red LED) - (((JoyStickY sensor value) - 512) / 100))
if <(3Color Red LED) > 255>
    set 3Color Red LED to 255
if <(3Color Red LED) < 0>
    set 3Color Red LED to 0
```

Need to be updated for Scratch 2

You need this block 3 times for the 3 colors

Part 3: The new counting function

Why a special counting function?

- You can count in Scratch, however you can only reliably count a few pulses per second.
- With this new counting function, you can do it much faster
- If you need to detect small pulses that scratchClient may not see reliable, you can use a counter function to detect whether a (short) pulse did arrive.

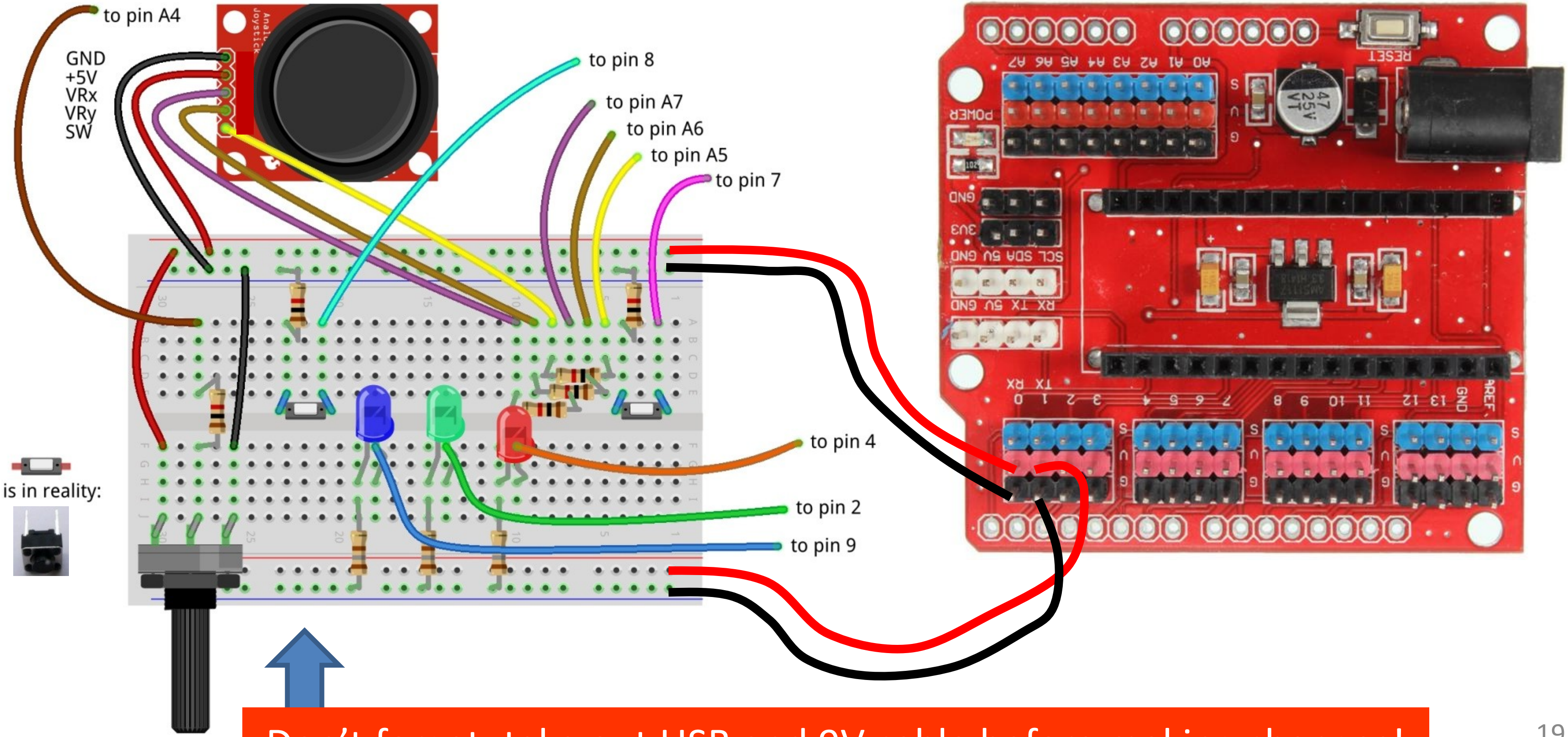
Counter function using a button

- It will count up. It will wrap round at a very large value that will be reached after days of counting.
- Max. ca. 80 counts per second = 4800 per minute
- There is a 4 ms debouncing delay
  - So multiple pulses within 4 ms will processed as a single count
  - So no need for capacitors to do debouncing

Debouncing

- If a mechanical switch is operated, it can generate a series of pulses rather than one.
- This is because one of the metal pieces bounces back a few times before it stays in position.
- You would normally need take care of this if you want to get a single pulse
  - E.g. add a capacitor.
- scratchClient will on the digital input pins only sample with 10 Hz, so it is unlikely that you will experience bounce problems.

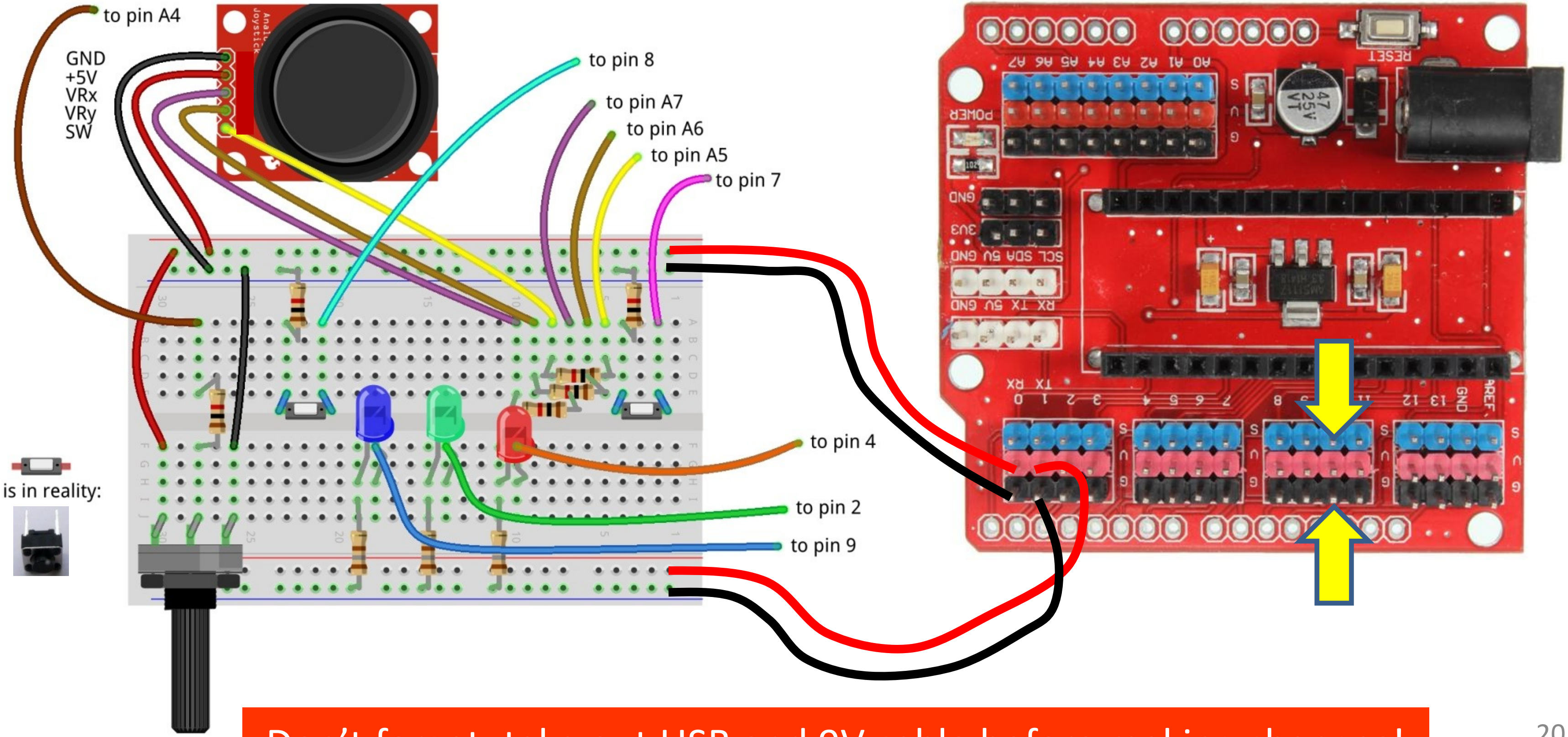
Adding the count button



Don’t forget: take out USB and 9V cable before making changes!

Adding an IR speed sensor that counts on pin 10

Carefully follow the wires to connect correctly!



Don’t forget: take out USB and 9V cable before making changes!

Differences between the counters

- The IR sensor gives an output of 0 Volt or 5 Volt.
  - Therefore it does not need a pull up resistor (see the config file)
- The button gives an output of 0 Volt or *open*.
  - Hence it needs a pull up resistor (as specified in the config file)

Try out the setup

- The button you can just press.
- The IR sensor you need to have good opaque material to let it work.
  - You can check the sensing by the LED at the backside.

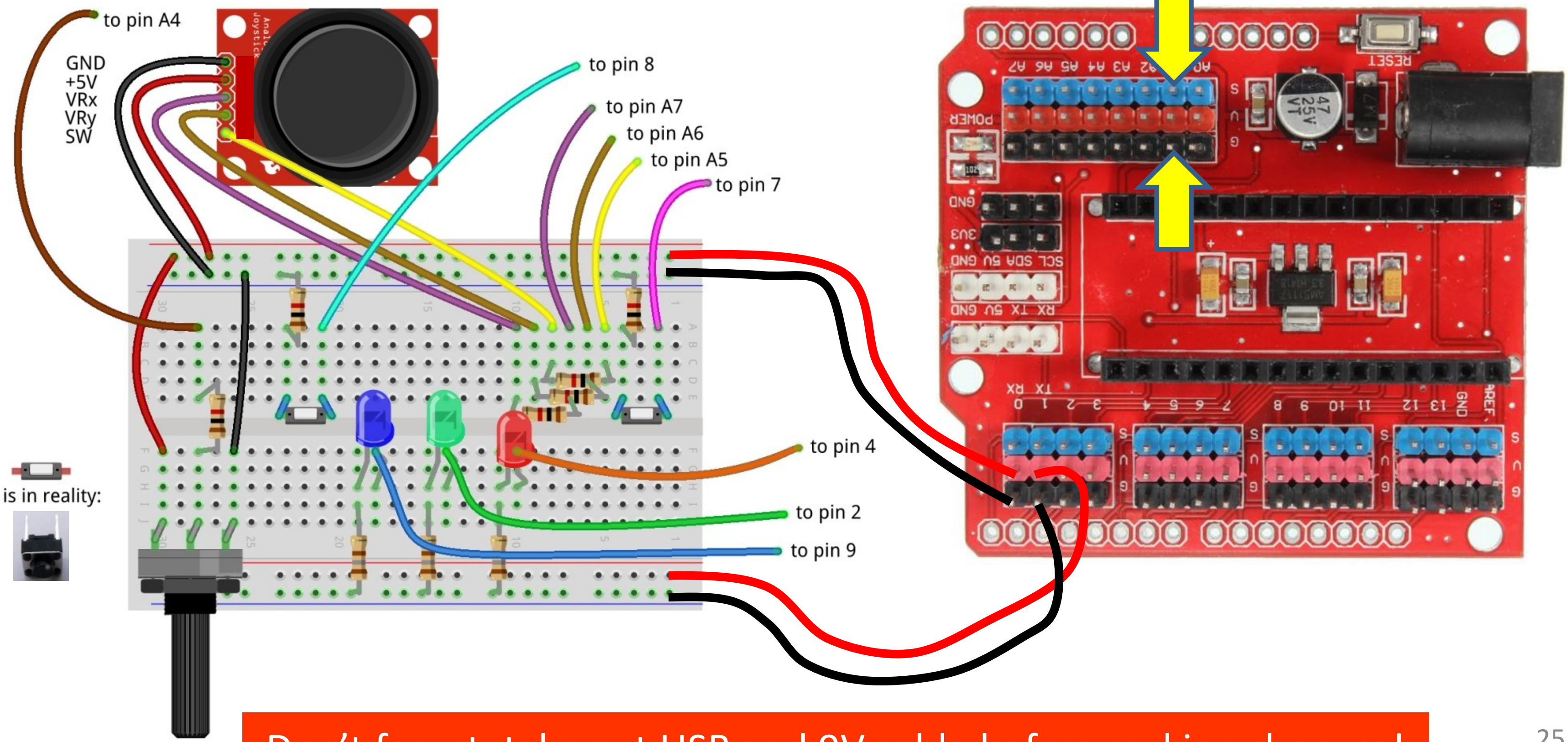
Part 4: Tilt sensor

Tilt sensor

- The tilt sensor is a sensor containing a rolling ball between contacts.
  - You can hear it rattling if you shake it.
- It is not in the box. We only have a few, so you will need to share.

Connect the tilt sensor to pin A1

Carefully follow the wires to connect correctly!



Don’t forget: take out USB and 9V cable before making changes!

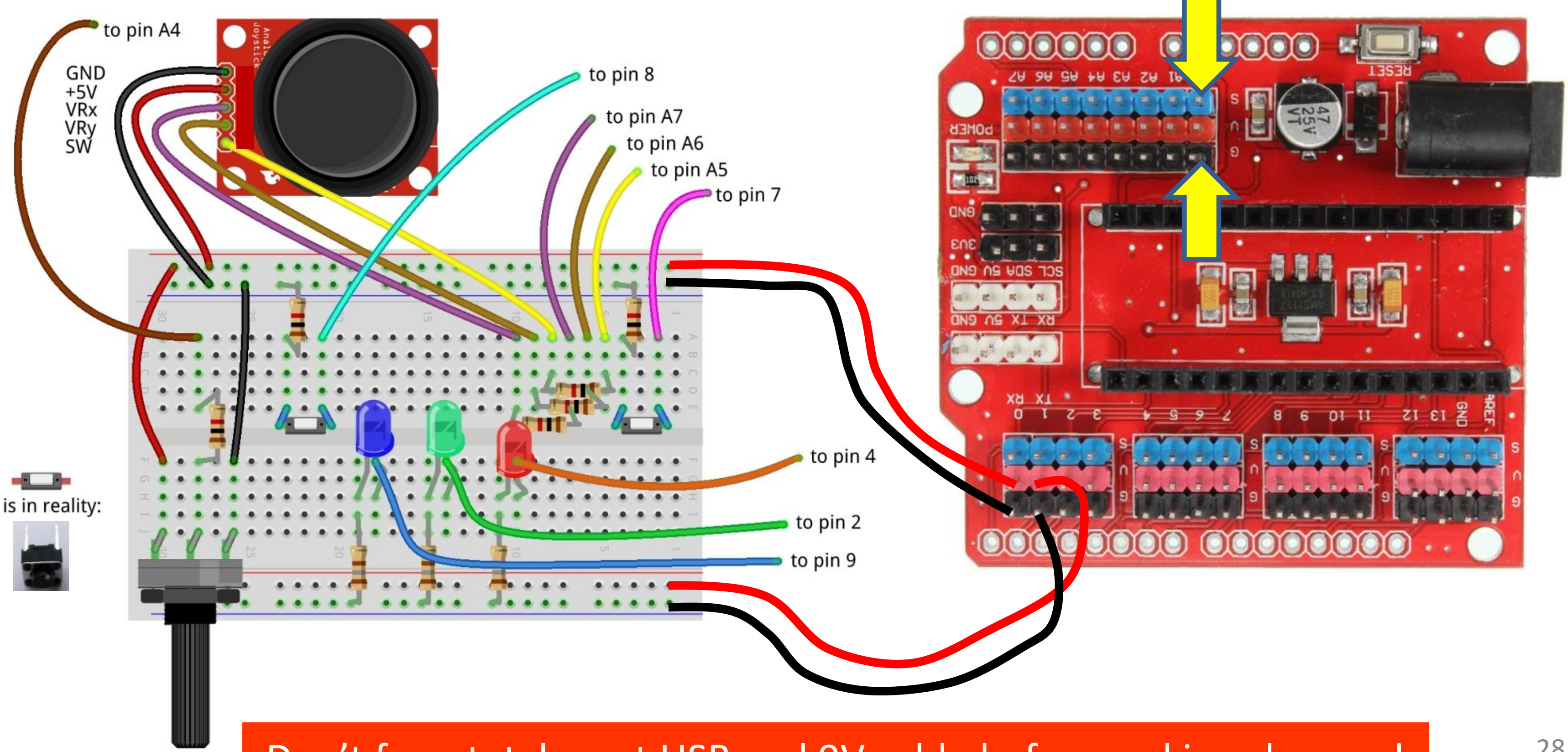
Part 5: Rain Sensor

Rain sensor

- A rain sensor is an analog sensor that measures conductivity that is influenced by water.
- It is not in the box. We only have a few, so you will need to share.

Connect the rain sensor to pin A0

Carefully follow the wires to connect correctly!



Don’t forget: take out USB and 9V cable before making changes!

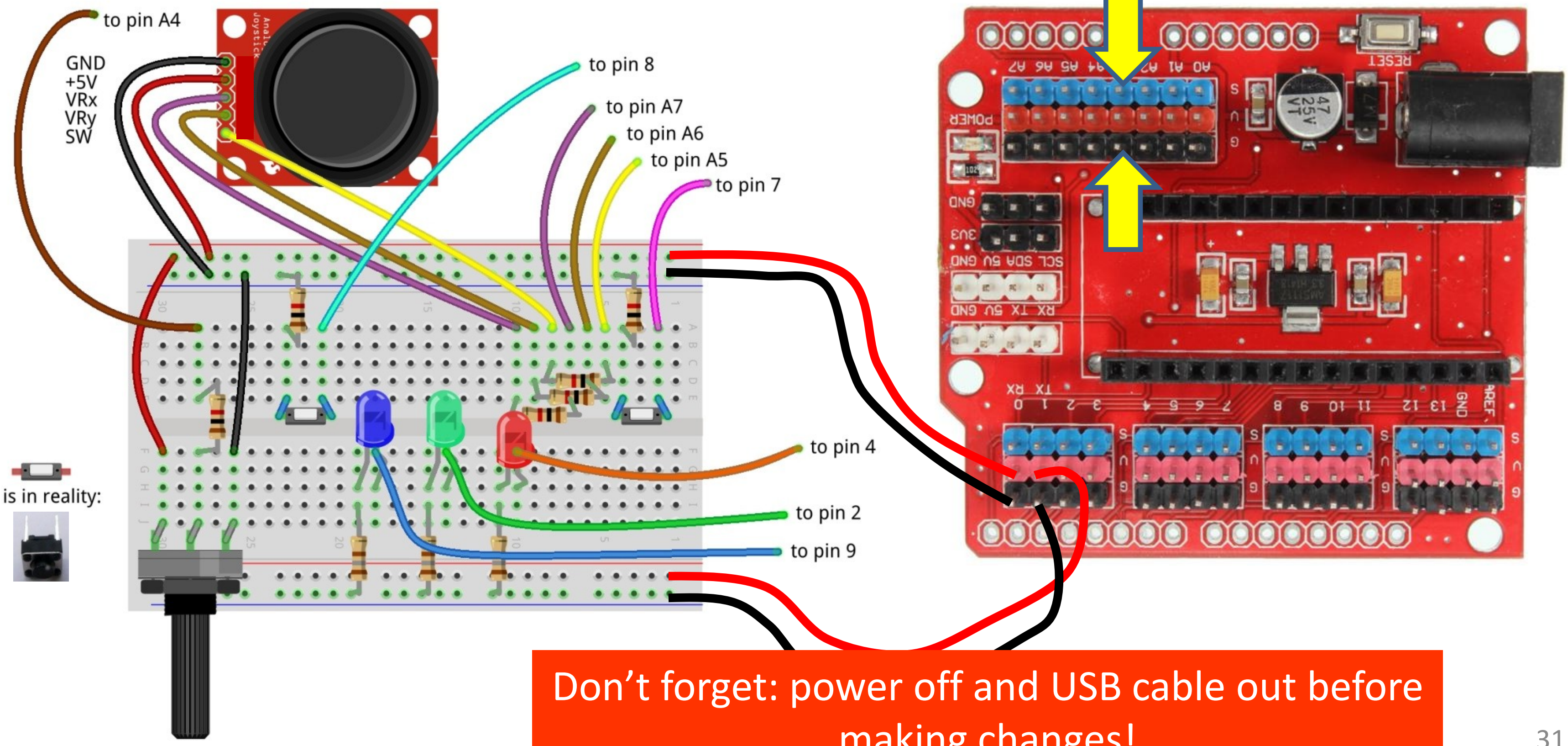
Part 6: Sound sensor

Sound Sensor

- The sound sensor will detect sound.
  - It is a digital sensor that contains a microphone and will give 0 Volt if sound is detected, 5 Volt if no sound detected.
- It is not in the box. We only have a few, so you will need to share.

Connect the sound sensor to pin A3

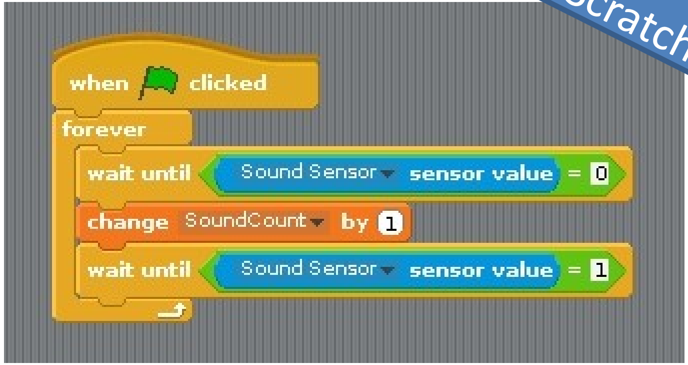
Carefully follow the wires to connect correctly!



Don’t forget: power off and USB cable out before making changes!

See the short pulses

- The changes of the sound sensor are sometimes difficult to see just from the displayed sensor value
- This code can be used to see that the sensor is actually working. You must create a variable SoundCount (or choose a different name) to hold the count.



```
when [flag] clicked
forever
    wait until < [Sound Sensor ▾] sensor value = 0 >
    change [SoundCount ▾] by 1
    wait until < [Sound Sensor ▾] sensor value = 1 >
```

Need to be updated for Scratch 2

The end of the **Advanced** Workshop